Virtual Mouse Using Hand Gesture Recognition

Mr. D.Suresh[1], Ponnien Selvan.N[2], ,Palanisamy.S[3], Ramprasath.T[4]
[1]professor, Dept of Computer Science and Engineering
[2, 3, 4]Dept of Computer Science and Engineering
[1, 2, 3, 4] PSNA College of Engineering and Technology, Dindigul, Tamil Nadu, India

Abstract- *Researchers from each over the world are working to make our bias more interactive and to make them work with minimum physical contact. In this study, we suggest an interactive computer system that can serve without the operation of a keyboard or a mouse. This device has the implicit to profit everyone, especially paralyzed people who have difficulty using a real mouse. Virtual Mouse with Hand Gesture Recognition is a design that shows a control mouse movement with a real- time camera/ Web camera. Our idea is to employ a camera and computer vision technologies to manage mouse tasks (clicking and scrolling), and we demonstrate how it can do all that mouse. This design demonstrates how to construct a mouse control system. The proposed system is made up of nothing further than a detector, which is a normal- resolution webcam that can follow the hand in two confines. OpenCV and Python will be used to make the system. Hand gestures are the most natural and royal manner of communicating. The camera's affair will be displayed on the examiner. It improves the recognition of mortal hand postures in a Human Computer Interaction operation, reduce the time spent calculating and the comfort related to mortal hand postures. Grounded on the proposed algorithm and named hand point, the operation has good time- grounded performance. And finds it easier to operate the system due to the proposed hand postures.*

Keywords- OpenCv, Mediapipe, CNN(Convolutional neural network),Gray level Co-Ocuurence matrix(GLCM)

I. INTRODUCTION

With the development technologies in the areas of stoked reality and bias that we use in our diurnal life, these bias are getting compact in the form of VR technology. This paper proposes an AI virtual mouse system that makes use of the hand gestures and hand tip discovery for performing mouse functions in the computer using computer vision. The main ideal of the proposed system is to perform computer mouse cursor functions and scroll function using a web camera or erected- in camera in the computer rather of using a traditional mouse device. Hand gesture and hand tip discovery by using computer vision is used as a HCI with the computer. With the use of the AI virtual mouse system, we can track the fingertip of the hand gesture by using a erected- in camera or web camera and perform the mouse cursor operations and scrolling function and also move the cursor with it. While using a wireless or a Bluetooth mouse, some bias similar as the mouse, the dongle to connect to the PC, and also, a battery to power the mouse to operate are used, but in this paper, the stoner uses his/ her erected- in camera or a webcam and uses his/ her hand gestures to control the computer mouse operations. In the proposed system, the web camera captures and also processes the frames that have been captured and also recognizes the various hand gestures and hand tip gestures and performs the particular mouse function. Python programming language is used for developing the AI virtual mouse system, and also, OpenCV which is the library for computer vision is used in the AI virtual mouse system. In the proposed AI virtual mouse system, the model makes use of the MediaPipe package for the shadowing of the hands and for shadowing of the tip of the hands, and also, Pynput, Autopy, and PyAutoGUI packages were used for moving around the window screen of the computer for performing functions analogous as left click, right click, and scrolling functions.

II. LITERATURESURVEY

There are some affiliated workshop carried out on virtual mouse using hand gesture discovery by wearing a glove in the hand and also using color tips in the hands for gesture recognition, but they're no more accurate in mouse functions. The recognition isn't so accurate because of wearing gloves; also, the gloves are also not suited for them, and in some cases, the recognition isn't so accurate because of the failure of discovery of color tips. Some sweats have been made for camera- grounded discovery of the hand gesture interface.

To overcome the stated problems, Zhengyou et al. (2001), proposed an interface system named Visual Panel that utilize arbitrary quadrangle-shaped planar object as a panel to allow the user to use any tip-pointer tools to interact with the computer. The interaction movements will be captured, analysed and implement the positions of the tip-pointer, resulting accurate and robust interaction with the computer. The overall system consists of panel tracker, tip-pointer tracker, holography, calculation and update, and action detector and event generator as it can simulate both mouse and

keyboard. However, although the proposed system solved the issues of cable length limitations, it still requires a certain area and material to operate. Zhengyou et al., have mentioned that the system can accepts any panel as long as it is quadrangle-shaped, meaning any other shape besides stated shape are not allowed.

Kamran Niyazi et al. (2012), mentioned that to solve the stated problem, ubiquitous computing method is required. Thus, colour tracking mouse simulation was proposed. The said system tracks two colour tapes on the user fingers by utilizing the computer vision technology. One of the tapes will be used for controlling the movement of the cursor while the other will act as an agent to trigger the click events of the mouse. To detect the colours, the system are first required to process the captured image by separating the hand pixels from the non-hand pixels, which can be done by background subtraction scheme that segments the hands movement information from the non-changing background scene. In order to implement this, the system requires to capture a pair of images to represent the static workplace from the camera view. When subtraction process is complete, the system will undergo another process that separates the RGB pixels to calculate the probability and differentiate the RGB values to determine which part are the skin and which are not. After the process is completed, it will start detecting the defined colour in the image, the image RGB pixels will be converted into HSV colour plane in order to eliminate the variation in shades of similar colour. The resulting image will be converted to Binary Image and will undergo a filtering process to reduce the noise within the image.

Even though the proposed system solved most of the stated issues, but there are limited functions offered by the proposed system as it merely able to perform common functions, such as: cursor movements, left/right click, and double clicks. While other functions, such as the middle click and mouse scroll were ignored.

Another colour detection method proposed by Kazim Sekeroglu (2010), the system requires three fingers with three colour pointers to simulate the click events. The proposed system are capable of detecting the pointers by referring the defined colour information, track the motion of the pointers, move the cursor according to the position of the pointer, and simulate the single and double left or/and right click event of the mouse.

To detect the colours, they have utilized the MATLAB's built in "imsubtract" function, with the combination of the noise filtering by using median filter, which are effective in filtering out or at least reduce the "salt and pepper" noise. The captured image will be converted to Binary Scale Image by using MATLAB's built in "im2bw" function to differentiate the possible values for each pixel. When the conversion is done, the captured image will undergo another filtering process by using "bwareaopen" to remove the small areas in order to get an accurate number of the object detected in the image.

Another "Ubiquitous Computing" approach proposed by Chu-Feng Lien (2015), requires only finger-tips to control the mouse cursor and click events. The proposed system doesn't requires hand-gestures nor colour tracking in order to interact with the system, instead it utilize a feature name Motion History Images(MHI) , a method that used to identify movements with a row of images in time.

Even though the proposed system possess good accuracy in a well-controlled environment, it does has its own limitations. The proposed system are not capable to detect fast moving movements as the frame-rates are not able to keep up, thus leading to increase of error rate. Furthermore, due to the mouse click events occurred when the finger hold on a certain positions, this may lead to user constant finger movements to prevent false alarm, which may result inconvenience.

III. EXISTINGSYSTEM:

The existing system consists of the general mouse and track pad system of monitor controlling and the non-availability of a hand gesture system. The remote penetrating of examiner screen using the hand gesture is unapproachable. Indeed though it's largely trying to apply the compass is simply confined in the field of virtual mouse. The being virtual mouse control system consists of the simple mouse operations like mouse pointer control, left click, right click, drag etc. The farther use of the hand recognition isn't been made use. Indeed still there are a number of systems, which are used for hand recognition, the system they made accustomed is the stationary hand recognition which is simply recognition of the shape made by hand and by defining an action for each shape made, which is limited to a number of defined conduct and a large amount of confusion.

DRAWBACKS

Limited accuracy: A virtual mouse relies on a touchpad or other touch-sensitive surface to track movement, which can be less accurate than a physical mouse. This can make precise movements difficult, which can be frustrating for tasks that require a high degree of accuracy.

Reduced speed: Virtual mice can be slower than physical mice, which can make tasks that require quick movements, such as gaming or graphic design, more challenging.

Ergonomic concerns: Using a virtual mouse for long periods can lead to hand and wrist strain, as well as other ergonomic issues. This is because virtual mice often require users to hold their hand in an unnatural position, which can cause discomfort over time.

Limited functionality: Some virtual mice may lack the additional buttons and features that are common on physical mice, which can limit their usefulness for certain tasks.

Power consumption: Using a virtual mouse on a portable device can consume battery power, which can reduce the device's overall battery life.

IV. PROPOSEDSYSTEM

A virtual mouse system typically allows users to control the movement of the cursor on a computer screen without the use of a physical mouse. Here are some components that could be part of a proposed system for a virtual mouse.

Hand detection: Hand detection is an important step in creating a virtual mouse system using hand gestures. Hand detection allows the system to identify the user's hand in the video stream and track its movement to control the cursor on the computer screen.

Background Subtraction: The next step is to subtract the background from the image to isolate the hand. This can be done using OpenCV background subtraction functions, such ascreateBackgroundSubtractorMOG2.

Skin Color Detection: The next step is to detect the skin color of the hand. This can be done using OpenCV in Range function, which can isolate the pixels with the skin color.

Contour Detection: The next step is to detect the contours of the hand. This can be done using OpenCV find Contours function,which can detect the boundaries of the hand.

Convex Hull: The next step is to find the convex hull of the hand. This can be done using OpenCV's convexHull function, which can find the smallest convex polygon that can contain all the points of the hand.

Hand tracking: This is an important component in creating a virtual mouse using hand gesture. Hand tracking allows the system to accurately detect the movement and position of the user's hand, which is used to control the cursor on the computer screen. Here are some techniques that can be used for hand tracking in a virtual mouse system

Feature-Based Tracking: This technique involves detecting and tracking certain features of the hand, such as fingertips or knuckles. Once these features are identified, they can be tracked over time to estimate the position and movement of the hand.

Color-Based Tracking: This technique involves tracking the movement of a certain color, such as a glove worn by the user. The system can use color thresholding and filtering techniques to isolate the hand and track its movement in real-time.

Depth-Based Tracking: This technique involves using depth sensors, such as the Kinect or Intel RealSense camera, to track the movement of the user's hand in 3D space. The depth data can be used to estimate the position and movement of the hand relative to the camera.

Optical Flow Analysis:This technique involves analyzing the movement of pixels in consecutive frames of a video stream to estimate the movement of the hand. This technique can work well in well-lit environments with clear and consistent backgrounds.

Gesture Recognition: Once the hand is tracked, the next step is to recognize different hand gestures. This can be done using machine learning algorithms, such as neural networks or decision trees. Common hand gestures that can be used as virtual mouse controls include swiping, clicking, and scrolling.

Cursor Control: Once a gesture is recognized, the module needs to translate it into cursor movements on the computer screen. This can be done using a mapping function that translates the gesture into specific cursor movements or actions.A virtual mouse system typically allows users to control the movement of the cursor on a computer screen without the use of a physical mouse. Here are some components that could be part of a proposed system for a virtual mouse.Controlling a virtual mouse using hand gestures can be achieved using various computer vision techniques, such as object detection, hand tracking, and gesture recognition.

Collect a dataset of hand gestures: You need to collect a dataset of hand gestures that you want to use to control the virtual mouse. You can record video footage of yourself

performing various hand gestures such as moving your hand up, down, left, right, and clicking.

Train a gesture recognition model: You need to train a machine learning model to recognize the hand gestures you collected. You can use deep learning techniques such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) to classify the hand gestures.

Implement hand tracking: Once you have a gesture recognition model, you need to track the position of the hand in the video feed. There are various algorithms available for hand tracking, such as the OpenCV hand tracking algorithm or the MediaPipe Hand Tracking solution.

Map hand gestures to mouse movements: Finally, you need to map the recognized hand gestures to mouse movements. For example, moving your hand to the right could move the virtual mouse to the right, while a click gesture could simulate a mouse click.

SYSTEM DESIGN:

The system architecture for a virtual mouse typically consists of the following components:

Input Device: This component captures the user's hand movements and sends them as input to the system. The input device can be a physical mouse or a touchpad, or even a camera that tracks the user's hand movements.

Capture the video stream: You would first need to capture the video stream from the camera using OpenCV's VideoCapture function.

Preprocess the video: Preprocessing the video involves converting it to grayscale and applying filters to remove noise and improve the image quality.

Detect the hand: You can use OpenCV's built-in hand detection algorithms to detect the hand in the video stream.
Track the hand: Once the hand has been detected, you can track its movements in real-time using OpenCV's tracking algorithms.

Translate hand movements into mouse movements: You can then translate the movements of the hand into movements of the virtual mouse cursor on the screen. For example, if the hand moves to the right, the cursor on the screen should move to the right as well.

Interpret hand gestures: Finally, you can interpret hand gestures such as clicks or swipes and use them to perform various actions on the computer.

Gesture Recognition Module: This component analyzes the input received from the input device and recognizes the hand gestures made by the user.

Collect and prepare the training data: Collect a dataset of images or videos of hand gestures that you want to recognize. Label each gesture with a unique identifier. Preprocess the data by cropping, resizing, and converting the images to grayscale.

Train the gesture recognition model: Use machine learning algorithms such as SVM, CNN, or Decision Trees to train the gesture recognition model. This involves splitting the data into training and validation sets, selecting the appropriate features, and tuning the hyperparameters.

Collect live video stream: Capture live video stream from the camera using OpenCV's VideoCapture function.

Preprocess the video stream: Preprocess the video by converting it to grayscale and applying filters to remove noise and improve the image quality.

Detect the hand: Detect the hand in the video stream using OpenCV's built-in hand detection algorithms.

Track the hand: Track the hand movements in real-time using OpenCV's tracking algorithms.

Recognize the gesture: Use the trained gesture recognition model to recognize the specific hand gestures. This involves analyzing the shape, size, and movement of the hand.

Translate hand movements into mouse movements: Translate the hand movements into movements of the virtual mouse cursor on the screen. For example, if the hand moves to the right, the cursor on the screen should move to the right as well.

Perform actions based on the recognized gesture: Interpret the recognized gestures and use them to perform various actions on the computer, such as left-click, right-click, or scroll.

Test and refine the system: Test the system with different users, lighting conditions, and hand positions to ensure that it works reliably.

Virtual Mouse Application: This component maps the recognized gestures to mouse actions, such as moving the cursor, clicking, scrolling, and dragging.

Capture the video stream: Capture the video stream from the camera using OpenCV's VideoCapture function.

Preprocess the video: Preprocess the video by converting it to grayscale and applying filters to remove noise and improve the image quality.

Hand detection: Detect the hand in the video stream using OpenCV's built-in hand detection algorithms.
Hand tracking: Track the hand movements in real-time using OpenCV's tracking algorithms.

Translate hand movements into mouse movements: Translate the hand movements into movements of the virtual mouse cursor on the screen. For example, if the hand moves to the right, the cursor on the screen should move to the right as well.

Perform mouse actions: Use hand gestures to perform mouse actions, such as left-click, right-click, or scroll.

Add additional features: Add additional features, such as drag and drop, zoom in and out, and multi-touch.

Test and refine the application: Test the application with different users, lighting conditions, and hand positions to ensure that it works reliably. Refine the application by adjusting the parameters and adding new features if necessary.

Operating System Interface: This component interacts with the operating system's mouse driver to translate the virtual mouse actions into system-level commands.

Capture the video stream: Capture the video stream from the camera using OpenCV's VideoCapture function.

Preprocess the video: Preprocess the video by converting it to grayscale and applying filters to remove noise and improve the image quality.

Hand detection: Detect the hand in the video stream using OpenCV's built-in hand detection algorithms.

Hand tracking: Track the hand movements in real-time using OpenCV's tracking algorithms.

Translate hand movements into mouse movements: Translate the hand movements into movements of the virtual mouse cursor on the screen. For example, if the hand moves to the right, the cursor on the screen should move to the right as well.

Perform mouse actions: Use hand gestures to perform mouse actions, such as left-click, right-click, or scroll.

Integrate with operating system: Use libraries and APIs provided by the operating system to perform various tasks. For example, use the Win32 API on Windows to control windows, menus, and dialogs. Use the AppleScript on macOS to automate tasks and control applications.

Implement voice commands: Use speech recognition libraries and APIs to implement voice commands that can control the operating system interface in conjunction with the hand gestures.

Test and refine the system: Test the system with different users, lighting conditions, and hand positionsto ensure that it works reliably.

Output Display: This component displays the virtual mouse cursor and the user's interactions with the virtual mouse on the screen.

Create a window: Create a window using OpenCV's namedWindow function.

Initialize the virtual mouse cursor position: Set the initial position of the virtual mouse cursor to the center of the window.

Draw the virtual mouse cursor: Draw the virtual mouse cursor on the screen using OpenCV's circle function. The size and color of the cursor can be adjusted according to your preference.

Draw the detected hand: Draw the detected hand on the screen

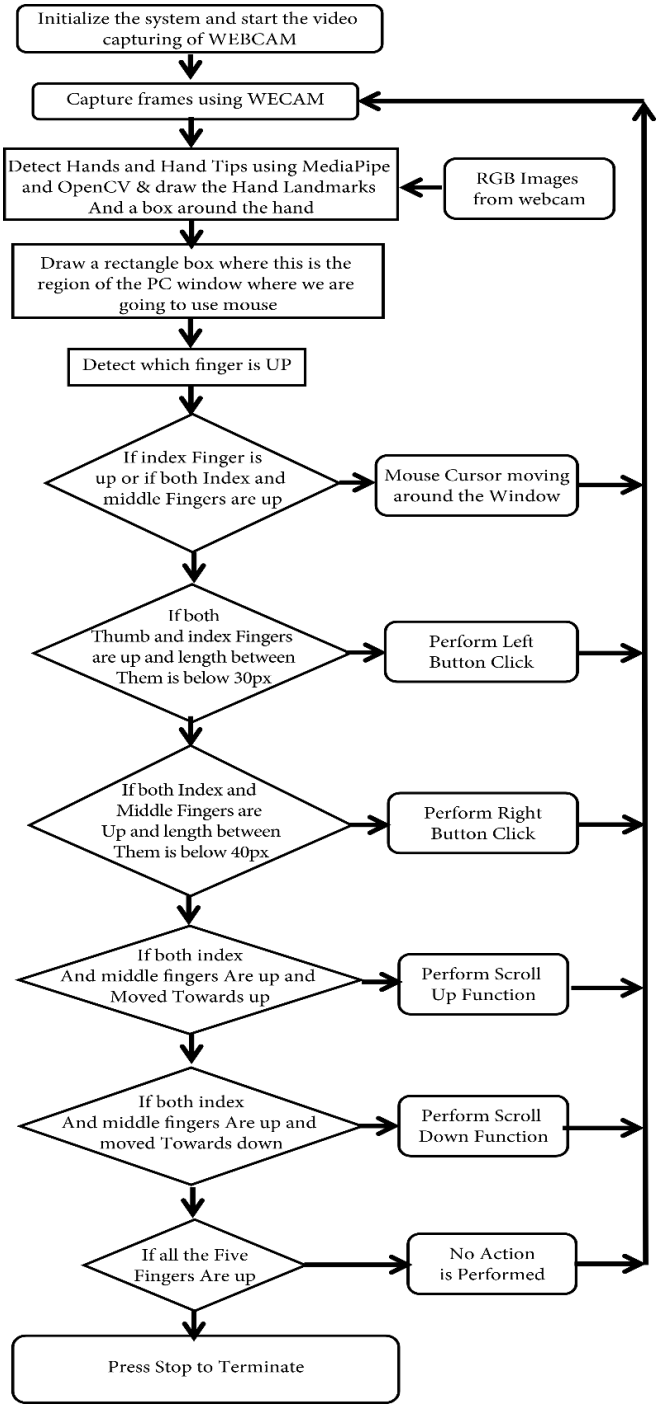

FIGURE 1:SYSTEM DESIGN

using OpenCV's rectangle or contour drawing functions. This can help the user to visualize their hand movements and ensure that the system is detecting the hand correctly.

Draw the mouse actions: Draw indicators on the screen to show the user which mouse actions are being performed, such as left-click, right-click, or scroll.

Display the output: Display the output on the screen using OpenCV's in show function. This function takes the name of the window and the image to be displayed as arguments.

Update the display: Update the display at a regular interval using OpenCV's waitKey function. This function waits for a key event or a timeout, and then updates the display accordingly.

Test and refine the application: Test the application with different users, lighting conditions, and hand positions to ensure that it works reliably. Refine the application by adjusting the parameters and adding new features if necessary. Overall, displaying the output in a Virtual Mouse Application with OpenCV in a virtual mouse using hand gesture is an important aspect of the user interface, as it provides visual feedback to the user and helps them to understand how the system is interpreting their hand movements. By drawing the virtual mouse cursor, detected hand, and mouse actions on the screen, you can create a more intuitive and engaging user experience.

V. CONCLUSION

In this design, we're working on a system to control the mouse cursor using a real- time camera. The system is predicated on computer vision algorithms and can do all Mouse tasks. still, it's delicate to get stable results because of the variety of lightning and skin colors of mortal races. It would be easier with this system, and work space would be saved by using it. It provides features similar as enlarging and shrinking Windows, closing window, etc. by using the Palm and multiple finger points. In order to make physically challenged people use desktops and laptops as smart as normal people, and consider the use of new edge technologies, this software is designed to make them as comfortable as the normal people. The main ideal of the virtual mouse system is to control the mouse cursor functions by using the hand gestures for reduce the usage of physical mouse. The proposed system can be achieved by using a webcam or a erected- in camera which detects the hand gestures and hand tip and processes these frames to perform the particular mouse functions. We've successfully enforced the cursor movement using the hand gesture. This is a design which uses the whole new technology making the mortal computer dealings in an easy and friendly way with a veritably minimum design cost.

VI. OUTPUT

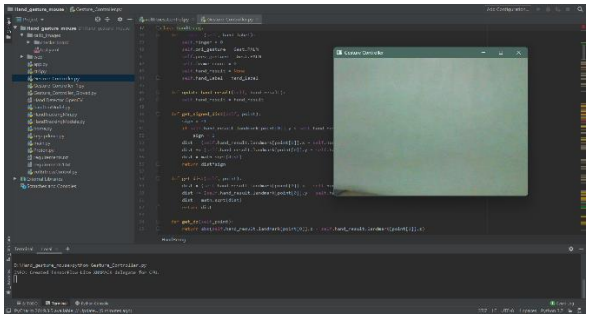

FIGURE 1:Accessing webcam

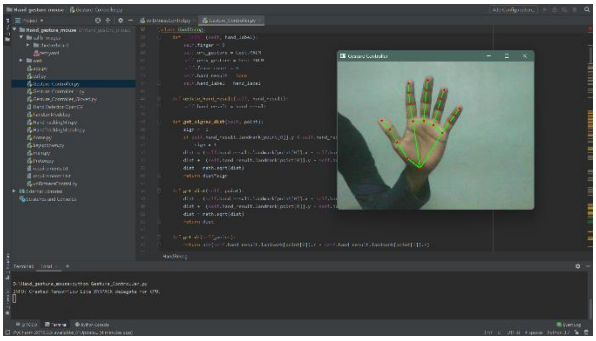

FIGURE 2:Hand detection

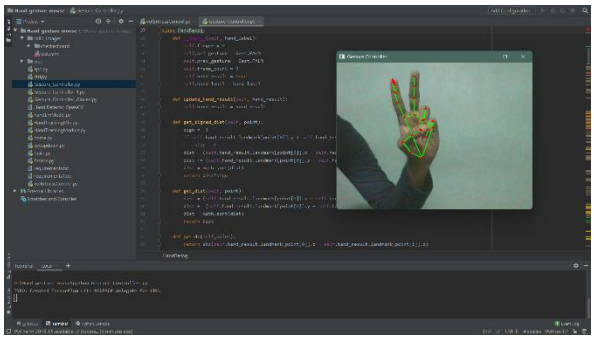

FIGURE 3:Cursor Movement

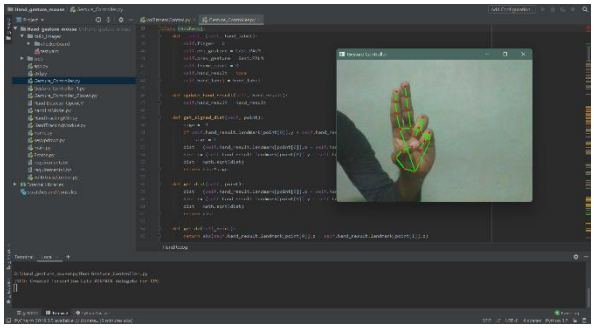

FIGURE 4:Double click

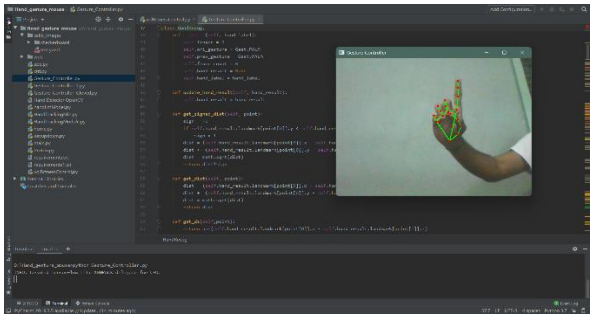

FIGURE 5:Single Click

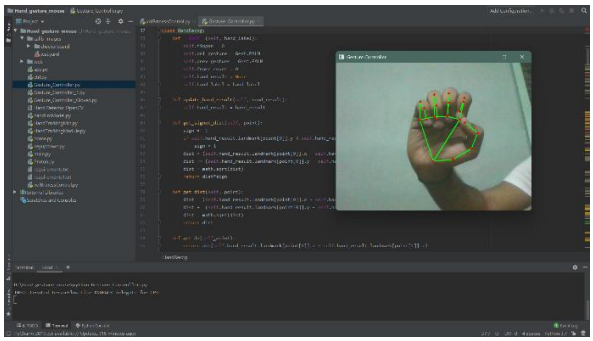

FIGURE 6:Drag and Drop

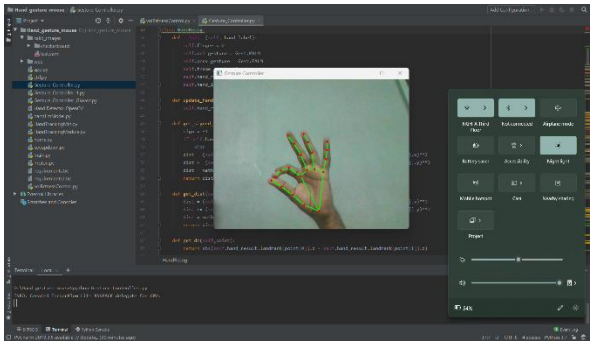

FIGURE 7:Accessing Volume and Brightness

REFERENCES

[1] Banerjee, A., Ghosh, A., Bharadwaj, K., &Saikia, H. (2014). Mouse control using a web camera based on colour detection. *arXiv preprint arXiv:1403.4722*.

[2] Chu-Feng,L.(2008).PortableVision-BasedHCI.[online]Availableat: http://www.csie.ntu.edu.tw/~p93007/projects/vision/vision_hci_p93922007.pdf [Accessed 25 Aug. 2015].

[3] Park, H. (2008). A method for controlling mouse movement using a real-time camera. *Brown University, Providence, RI, USA, Department of computer science*.

[4] Kumar N, M. (2011). *Manual Testing: Agile software development*. [online] Manojforqa.blogspot.com. Availableat: http://manojforqa.blogspot.com/2011/09/agile-software-development.html [Accessed 27 Aug. 2015].

[5] Niyazi, K. (2012). Mouse Simulation Using Two Coloured Tapes. *IJIST*, 2(2), pp.57- 63.
[6] Sekeroglu, K. (2010). Virtual Mouse Using a Webcam. [online]Availableat: http://www.ece.lsu.edu/ipl/SampleStudentProjects/ProjectKazim/Virtual%20Mouse%20Using%20a%20Webcam_Kazim_Sekeroglu.pdf [Accessed 29 Aug. 2015].
[7] Tutorialspoint.com, (n.d.). *SDLC - Agile Model*. [online] Available at: http://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm [Accessed 27 Aug. 2015].
[8] Tabernae.com, (n.d.). *Software Life Cycle | Web Development Outsourcing| IT Offshore Outsourcing*. [online] Available at: http://www.tabernae.com/process.aspx [Accessed 28 Aug. 2015].
[9] Zhengyou, Z., Ying, W. and Shafer, S. (2001). Visual Panel: Virtual Mouse, Keyboard and 3D Controller with an Ordinary Piece of Paper. [online] Available at: http://research.microsoft.com/en-us/um/people/zhang/Papers/PUI2001- VisualPanel.pdf [Accessed 25 Aug. 2015].